Data Compression
Lecture Notes

Dr. Faisal Aslam

Lecture 1 Introduction to Data Compression

1.1 Learning Objectives

By the end of this lecture, students will be able to:

- Define data compression and explain its practical importance with real-world examples
- Differentiate between lossless and lossy compression with concrete applications
- Calculate and interpret basic compression metrics (compression ratio, bit-rate, savings)
- Explain the concepts of information, redundancy, and entropy with computational examples
- Identify major application domains and their specific compression requirements
- Understand the fundamental limits of compression from information theory

1.2 Introduction and Motivation: Why Compress Data?

Data compression is the process of encoding information using fewer bits than the original representation. Every day, we encounter compression without realizing it: from streaming videos to sending emails, from saving photos to downloading software updates.

Definition

Data Compression: The process of reducing the number of bits needed to represent information, while either:

- **Lossless**: Preserving all original information exactly
- **Lossy**: Accepting some controlled loss of information for higher compression

1.2.1 Real-World Motivation: A Concrete Example

Consider a typical smartphone photo: 12 megapixels, 24-bit color (8 bits per RGB channel). Uncompressed size:

$$12,000,000 \text{ pixels} \times 24 \text{ bits/pixel} = 288,000,000 \text{ bits} = 36 \text{ MB}$$

But your phone stores it as a 3 MB JPEG file. That's a 12:1 compression ratio! Without compression:

- Your 128 GB phone could store only 3,500 photos instead of 40,000
- Uploading to social media would take 12 times longer
- Cloud storage costs would be 12 times higher

1.2.2 The Economics of Compression

Example

Cloud Storage Example: A major cloud provider charges \$0.023 per GB per month. For 1 PB (petabyte = 1000 TB) of data:

- Uncompressed: 1 PB = 1,000,000 GB gives \$23,000/month
- With 4:1 compression: 250,000 GB gives \$5,750/month
- Annual savings: (\$23,000 - \$5,750) $\times$ 12 = \$207,000/year

This doesn't even consider bandwidth costs, which are typically charged per GB transferred!

1.3 Lossless vs. Lossy Compression: A Detailed Comparison

1.3.1 Lossless Compression: Perfect Reconstruction

How it works: Exploits statistical redundancy and patterns without losing information.

Key techniques:

1. **Entropy coding**: Assign shorter codes to frequent symbols (Huffman, Arithmetic)
2. **Dictionary methods**: Replace repeated patterns with references (LZ77, LZ78)
3. **Predictive coding**: Encode differences from predictions rather than raw values

Example

Text Compression Example: The word "compression" appears 100 times in a document.

- Uncompressed: "compression" = 11 characters × 8 bits = 88 bits × 100 = 8,800 bits
- Compressed: Assign code "01" (2 bits) for "compression" → 2 bits × 100 = 200 bits
- Plus dictionary entry: "compression" = 88 bits (stored once)
- Total: 200 + 88 = 288 bits vs 8,800 bits → 30:1 compression!

This is essentially how LZW (used in GIF, ZIP) works.

1.3.2 Lossy Compression: Intelligent Approximation

How it works: Removes information that is:

- Imperceptible to humans (psychovisual/psychoacoustic models)
- Less important for the intended use
- Redundant beyond a certain quality threshold

Example

JPEG Image Compression - Step by Step:

1. **Color Space Conversion**: RGB to YCbCr (separates luminance from color)
2. **Chrominance Downsampling**: Reduce color resolution (4:2:0) - humans are less sensitive to color details
3. **Discrete Cosine Transform (DCT)**: Convert 8×8 pixel blocks to frequency domain
4. **Quantization**: Divide frequency coefficients by quantization matrix - small high-frequency coefficients become zero
5. **Entropy Coding**: Huffman code the results

   Result: Typical 10:1 to 20:1 compression with minimal visible artifacts

1.3.3 When to Use Which? Decision Factors

The choice between lossless and lossy compression depends on the acceptable level of information loss, the nature of the data, and system constraints such as speed and storage. Lossless compression is required whenever exact reconstruction is mandatory, whereas lossy compression is preferred when human perception can tolerate approximations in exchange for significantly higher compression ratios. The following table summarizes the key decision factors commonly encountered in practice.

Factor	**Choose Lossless When**	**Choose Lossy When**
Fidelity Requirement	Exact reconstruction is critical (code, financial data, legal documents)	Some quality loss is acceptable (media streaming, web images)
Data Type	Discrete data with exact values (text, databases, executables)	Continuous data with perceptual limits (images, audio, video)
Compression Ratio Needed	Moderate ratios suffice (2:1 to 10:1)	High ratios needed (10:1 to 200:1+)
Processing Requirements	Fast decompression needed, encode speed less critical	Real-time encoding/decoding needed (streaming, videoconferencing)
Regulatory Constraints	Legal/medical requirements mandate exact copies	No regulatory constraints on quality

Table 1: Decision Factors for Lossless vs. Lossy Compression

1.4 Performance Metrics: Beyond Simple Ratios

1.4.1 Compression Ratio and Savings

$$\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}}$$

$$\text{Savings} = \left(1 - \frac{\text{Compressed Size}}{\text{Original Size}}\right) \times 100\%$$

Example

Comparing Different Compression Scenarios:

Scenario	**Original**	**Compressed**	**Ratio**	**Savings**
Text document (ZIP)	1.5 MB	450 KB	3.33:1	70%
CD Audio (FLAC lossless)	700 MB	350 MB	2:1	50%
Same Audio (MP3 128kbps)	700 MB	112 MB	6.25:1	84%
4K Video (H.265)	100 GB	2 GB	50:1	98%
DNA sequence (specialized)	3 GB	300 MB	10:1	90%

1.4.2 Bit-rate: The Quality Control Knob

For lossy compression, bit-rate determines quality:

$$\text{Bit-rate} = \frac{\text{Compressed Size in bits}}{\text{Duration (seconds)}} \quad \text{or} \quad \frac{\text{Compressed Size in bits}}{\text{Number of samples}}$$

Example

Audio Quality at Different Bit-rates:

- **32 kbps**: Telephone quality, speech only
- **96 kbps**: FM radio quality
- **128 kbps**: "Good enough" for most listeners
- **192 kbps**: Near CD quality for most people
- **320 kbps**: Essentially transparent (FLAC: 900 kbps)

  Storage impact: A 60-minute album:

  - At 128 kbps: 60 MB
  - At 320 kbps: 144 MB
  - FLAC lossless: 400 MB
  - Uncompressed CD: 700 MB

1.4.3 Time and Space Trade-offs

Compression involves multiple competing objectives:

$$\text{Space–Time Trade-off} = \frac{\text{Compression Ratio}}{\text{Encoding Time} \times \text{Decoding Time}}$$

While higher compression ratios reduce storage and bandwidth, they often come at the cost of increased computational complexity and latency. In real-time and large-scale streaming systems (e.g., Netflix, YouTube, Facebook), **encoding and especially decoding speed are often more critical than optimal compression ratios**. Streaming workloads require fast, low-latency decoding on a wide range of devices, from mobile phones to smart TVs, where CPU, memory, and power budgets are limited.

As a result, practical streaming systems favor compressors that achieve a *good enough* compression ratio while providing high throughput, low memory usage, and predictable latency, even if better compression is theoretically possible.

Example

Real-world Compressor Comparison:

Algorithm	Ratio (text)	Encode Speed	Decode Speed	Memory
gzip (-6)	3.2:1	100 MB/s	400 MB/s	10 MB
bzip2 (-6)	3.8:1	20 MB/s	50 MB/s	50 MB
LZ4	2.5:1	500 MB/s	2000 MB/s	1 MB
Zstd (-3)	3.0:1	300 MB/s	800 MB/s	5 MB
xz (-6)	4.2:1	10 MB/s	80 MB/s	100 MB

Approximate performance on typical text data (higher is better)

1.5 Information and Redundancy: The Core Concepts

1.5.1 Information: Quantifying Surprise

In everyday language, information is often confused with the mere presence of data. In information theory, however, information measures how much an observation *reduces uncertainty*. If an outcome is fully predictable, observing it provides little or no new information.

Claude Shannon's key insight was that information is inversely related to probability: unlikely events are more informative because they are more surprising, while highly likely events convey little new knowledge.

Definition

Information Content of an event with probability p is defined as:

$$I(p) = -\log_2 p \quad \text{bits}$$

This definition captures an important intuition: as an event becomes more predictable ($p \to 1$), its information content approaches zero.

Example

Predictability vs. Information:

- In a city where it rains every day, the statement "It rained today" conveys almost no information because it was already expected.
- A file that contains only the bit '1' provides very little information, since after seeing a few bits, the rest of the file can be predicted with certainty.
- A coin that always lands heads produces outcomes, but no information, because there is no uncertainty to resolve.

Key idea: Perfect predictability implies zero information gain.

Example

Daily Weather Forecast — Information Content:

- Sunny in Phoenix (probability 0.9): $I = -\log_2 0.9 \approx 0.15$ bits
- Snow in Phoenix (probability 0.001): $I = -\log_2 0.001 \approx 9.97$ bits
- Rain in Seattle (probability 0.3): $I = -\log_2 0.3 \approx 1.74$ bits

Interpretation: Rare events carry more information because they reduce uncertainty the most. Snow in Phoenix reveals far more about the weather system than another sunny day.

In summary, information is not about how much data is observed, but about how much uncertainty is removed. This distinction between information and predictability forms the foundation of redundancy, entropy, and data compression.

1.5.2 Redundancy: The Enemy of Information and the Friend of Compression

Redundancy refers to predictable or repeated structure in data. From the perspective of information theory, redundancy is the *enemy of information* because it does not reduce uncertainty. However, from the perspective of data compression, redundancy is a *valuable resource*: it is precisely what allows data to be represented using fewer bits.

Compression algorithms work by identifying, modeling, and removing redundancy while preserving the underlying information (in lossless compression) or perceptually important information (in lossy compression).

Redundancy appears in several common forms:

1. **Spatial Redundancy**: Neighboring data values are highly correlated.

Example

In a photograph of a clear blue sky, most neighboring pixels have nearly identical color values.

- **Naive**: Store the RGB value of each pixel independently.
- **Smarter**: Encode repeated pixel values using run-length encoding.
- **Even smarter**: Predict each pixel from its neighbors and encode only the small prediction error.

2. **Statistical Redundancy**: Some symbols occur far more frequently than others.

Example

English letter frequencies:

Letter	Frequency	Letter	Frequency
E	12.7%	Z	0.07%
T	9.1%	Q	0.10%
A	8.2%	J	0.15%

- **Inefficient**: Fixed-length coding (5 bits per letter).
- **Efficient**: Variable-length coding (e.g., Huffman coding), where frequent letters get shorter codes.

This reduces the average bits per letter from 5 to approximately 4.1.

3. **Knowledge Redundancy**: Information already known to both encoder and decoder.

Example

Medical Imaging: Both the encoder and decoder know the image represents a chest X-ray.

- The general structure of lungs and bones does not need to be encoded explicitly.
- Anatomical models can be used to predict expected structures.
- Bits can be concentrated on unexpected or diagnostically important regions.

4. **Perceptual Redundancy**: Information that humans cannot perceive.

Example

Audio Compression (MP3):

- **Frequency masking**: Loud sounds mask nearby frequencies.
- **Temporal masking**: Loud sounds mask quieter sounds before or after them.
- **Result**: A large fraction of the audio data can be discarded without perceptible loss in quality.

1.5.3 What is Entropy? Different Perspectives

The term "entropy" appears in multiple fields (thermodynamics, information theory, statistics) with related but distinct meanings. In information theory, we primarily discuss **Shannon Entropy**, named after Claude Shannon who founded the field in 1948. While there are other entropy measures (like Kolmogorov-Sinai, Rényi, and Tsallis entropies in various contexts), Shannon entropy is the foundational concept for data compression.

Definition

Shannon Entropy of a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$ having probabilities $\{p_1, p_2, \ldots, p_n\}$:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad \text{bits}$$

Two Complementary Interpretations:

1. **Average Information Content**: When a symbol with probability p_i occurs, it conveys $-\log_2 p_i$ bits of information (rare events tell us more). Entropy is the *expected value* or average of this information content across all symbols.

2. **Uncertainty or Surprise**: Entropy measures how uncertain we are about the next symbol before observing it. Higher entropy means more unpredictability.

These interpretations are two sides of the same coin: *The average information gained equals the uncertainty removed by observation.*

1.5.4 Calculating Entropy: Step by Step

Let's examine both interpretations through detailed calculations:

Example

Binary Source Example - Detailed Calculation:

Consider a biased coin: P(Heads) = 0.8, P(Tails) = 0.2

Step 1: **Calculate individual information content**:

$$I_H = -\log_2(0.8) \approx 0.3219 \text{ bits}$$
$$I_T = -\log_2(0.2) \approx 2.3219 \text{ bits}$$

Interpretation: Tails (rarer event) carries more information.

Step 2: **Calculate entropy as expected value**:

$$H = 0.8 \times 0.3219 + 0.2 \times 2.3219 = 0.7219 \text{ bits}$$

Step 3: **Verify using direct formula**:

$$H = -[0.8 \log_2(0.8) + 0.2 \log_2(0.2)] \approx 0.7219 \text{ bits}$$

Key Insights:

- **Average information**: Each flip gives 0.72 bits of information on average
- **Uncertainty**: We're 72% as uncertain as with a fair coin
- **Extreme cases**:
  - Fair coin (P=0.5): $H = 1.0$ bit (maximum uncertainty/information)
  - Always heads (P=1.0): $H = 0$ bits (no uncertainty, no information)
  - 90% heads: $H \approx 0.469$ bits (less uncertainty than 80% case)

Mathematical Properties of Entropy:

- **Non-negativity**: $H(X) \geq 0$, with equality only when one outcome is certain
- **Maximum value**: For n symbols, maximum entropy is $\log_2 n$, achieved when all probabilities are equal ($p_i = 1/n$)
- **Concavity**: Entropy is a concave function of probabilities

1.5.5 Entropy of English Text: A Practical Case Study

Example

Calculating English Letter Entropy:

Based on letter frequencies in typical English text:

Letter	**Probability (p_i)**	$-\log_2 \mathbf{p_i}$	**Contribution ($p_i \times -\log_2 p_i$)**
E	0.127	2.98	0.378
T	0.091	3.46	0.315
A	0.082	3.61	0.296
O	0.075	3.74	0.281
I	0.070	3.84	0.269
N	0.067	3.90	0.261
S	0.063	3.99	0.251
H	0.061	4.04	0.246
R	0.060	4.06	0.244
D	0.043	4.54	0.195
⋮	⋮	⋮	⋮
Z	0.0007	10.48	0.007
Total	1.0		**4.18 bits**

Layered Interpretation:

- **First-order entropy (letters independent)**: 4.18 bits/letter
- **Why not 5 bits?** Because letters are not equally likely
- **Actual uncertainty is lower**: Letters have dependencies (Q is usually followed by U)
- **Comparison with encoding schemes**:

Encoding Method	**Bits/Letter**	**Efficiency**
Naive (5 bits for 26 letters)	5.00	83.6%
Huffman (letter-based)	4.30	97.2%
Using digram frequencies	3.90	107.2%*
Using word frequencies	2.30	181.7%*
Optimal with full context	1.50	278.7%*

*Percentages ¿100% show compression better than first-order entropy by exploiting dependencies

1.5.6 Beyond First-Order Entropy: The Full Picture

Higher-Order Entropy accounts for dependencies between symbols:

- **Zero-order entropy**: $H_0 = \log_2 n$ (equal probabilities)
- **First-order entropy**: $H_1 = -\sum p_i \log_2 p_i$ (letter frequencies only)
- **Second-order entropy**: $H_2 = -\sum p(i,j) \log_2 p(i|j)$ (letter pairs)
- **N-th order entropy**: $H_N = -\sum p(\text{block}) \log_2 p(\text{last}|\text{previous})$

The Entropy Rate is the limit as $N \to \infty$:

$$H_\infty = \lim_{N\to\infty} H_N$$

For English, $H_\infty \approx 1.0 - 1.5$ bits/letter, much lower than first-order entropy!

1.5.7 The Entropy Theorem: Why It Matters

Important

Shannon's Source Coding Theorem (Formal Statement):
Given a discrete memoryless source with entropy H, for any $\epsilon > 0$:

1. **Converse**: No code can have average length $L < H$ without losing information
2. **Achievability**: There exists a code with average length $L < H + \epsilon$

Implications for Compression:

- **Fundamental limit**: H is the absolute lower bound for lossless compression
- **Redundancy**: Difference between actual code length and H is wasted space
- **Optimality**: Good compressors approach H from above

Practical Example - English Text Compression:

- **Impossible**: Average $<$ 1.5 bits/letter (entropy rate limit)
- **Wasteful**: 8 bits/letter (ASCII - 533% of optimal)
- **Good**: 2.5 bits/letter (modern compressors - 167% of optimal)
- **Theoretical limit**: 1.5 bits/letter (100% efficiency)

Why We Can't Reach The Limit Exactly:

- Finite block sizes in practical codes
- Computational complexity of optimal coding
- Need for integer-length codes (Huffman)
- Model inaccuracy in estimating probabilities

Takeaway Message:

- **Entropy IS both**: average information AND uncertainty
- **First-order** H gives a baseline, but real sources have lower entropy rates
- **The theorem** tells us what's possible and impossible
- **Good compression** = modeling dependencies to approach H_∞

1.6 Application Domains: Specialized Requirements

1.6.1 Text and Code Compression

- **Requirements**: Lossless, fast random access, incremental updates
- **Challenges**: Small files, need for searching within compressed data
- **Solutions**: gzip (DEFLATE), LZ4, Zstandard

Example

Git Version Control: Uses zlib (DEFLATE) and delta compression:

- Stores file versions as compressed objects
- Applies delta compression for similar versions (packfiles)
- Exploits low *conditional entropy* between revisions
- Example: Linux kernel repository: ∼4 GB raw, ∼1 GB stored

1.6.2 Multimedia Compression

- **Requirements**: High compression, perceptual quality, real-time
- **Challenges**: Massive data volumes, human perception constraints
- **Solutions**: JPEG, MP3, H.264/HEVC, AV1

Example

Streaming Service Economics (Netflix/YouTube):

- 1 hour of 4K video: Uncompressed 500 GB
- H.265 compressed: 4 GB (125:1 compression)
- Bandwidth cost: \$0.05/GB (typical CDN pricing)
- Uncompressed stream: \$25/hour
- Compressed stream: \$0.20/hour
- For 100 million hours/day: \$20M/day vs \$2.5B/day!

1.6.3 Scientific and Medical Data

- **Requirements**: Lossless or controlled loss, reproducibility, standards

- **Challenges**: Huge datasets, precision requirements, regulatory compliance
- **Solutions**: Specialized compressors (SZ, ZFP), format standards (DICOM)

Example

Large Hadron Collider (LHC) Data:

- Generates 1 PB/second (yes, per second!)
- Stores 50 PB/year after filtering
- Uses specialized compression algorithms
- Compression saves $50M/year in storage costs
- Enables global collaboration (data distributed worldwide)

1.7 The Compression Pipeline: How Compressors Actually Work

Most compressors follow this two-stage process:

Compression Pipeline:

Input Data $\longrightarrow$ **Modeling Stage** $\longrightarrow$

Analyzes patterns Builds probability model

$\longrightarrow$ **Coding Stage** $\longrightarrow$

Converts to bits Using entropy coding

$\longrightarrow$ **Compressed Data**

Two-Stage Compression Pipeline:

- **Modeling Stage**: Analyzes data patterns and builds probability model
- **Coding Stage**: Converts symbols to bits using entropy coding (Huffman, Arithmetic, ANS)

1.7.1 Modeling Strategies in Practice

Example

Huffman Coding Example - Complete Process:

1. **Modeling**: Count symbol frequencies in "ABRACADABRA"

Symbol	Frequency	Probability
A	5	$5/11 \approx 0.455$
B	2	$2/11 \approx 0.182$
R	2	$2/11 \approx 0.182$
C	1	$1/11 \approx 0.091$
D	1	$1/11 \approx 0.091$

2. **Coding**: Build Huffman tree (simplified):
   - Combine lowest frequencies: C(1) + D(1) = CD(2)
   - Continue combining: CD(2) + B(2) = CDB(4)
   - Combine: CDB(4) + R(2) = CDBR(6)
   - Final: CDBR(6) + A(5) = Root(11)

3. **Code assignment**:

Symbol	Code	Length
A	0	1 bit
R	10	2 bits
B	110	3 bits
C	1110	4 bits
D	1111	4 bits

4. **Compress "ABRACADABRA"**:
   - A(0) B(110) R(10) A(0) C(1110) A(0) D(1111) A(0) B(110) R(10) A(0)
   - Total bits: 1+3+2+1+4+1+4+1+3+2+1 = 23 bits
   - Original: 11 characters $\times$ 8 bits = 88 bits
   - Compression: 88 $\rightarrow$ 23 bits (3.8:1 ratio)
   - Entropy limit: $H \approx 2.04$ bits/char $\times$ 11 = 22.5 bits
   - Efficiency: 22.5/23 = 97.8% efficient!

1.8 Important Terminology and Concepts

1.8.1 Key Definitions with Examples

- **Symbol**: The basic unit being compressed

  Example

  Different domains use different symbols:

  - Text: Characters (bytes)
  - Images: Pixels (RGB triples)
  - Audio: Samples (16-bit integers)
  - Video: Macroblocks (16×16 pixel regions)

- **Alphabet**: Set of all possible symbols

  Example

  - English text: 256 possible bytes (ASCII/UTF-8)
  - Binary data: 256 possible byte values
  - DNA sequences: 4 symbols {A, C, G, T}
  - Black-white image: 2 symbols {0=black, 1=white}

- **Prefix Code**: Crucial for instant decoding

  Example

  Why prefix codes matter:

  - Good: A=0, B=10, C=110, D=111
  - "010110" decodes unambiguously: A(0) B(10) C(110)
  - Bad: A=0, B=1, C=01 (not prefix-free)
  - "01" could be AB or C - ambiguous!

1.8.2 The Fundamental Insight

Important

The Core Principle of Compression:

- **Random data cannot be compressed**: Maximum entropy = no redundancy
- **Real-world data is not random**: Contains patterns, structure, predictability
- **Compression finds and exploits these patterns**

  Example - Encryption vs Compression:

  - Encrypted data looks random (high entropy)
  - Compressing encrypted data gives little or no savings
  - Always compress **before** encrypting, not after!
  - Rule: Encrypt $\rightarrow$ High entropy $\rightarrow$ No compression
  - Rule: Compress $\rightarrow$ Lower entropy $\rightarrow$ Then encrypt

1.9 Homework Assignment: Practical Exercises

1. **Compression Calculation**:
   - A 4K video frame is 3840×2160 pixels, 24-bit color. Calculate:
     (a) Uncompressed size in MB
     (b) Size after 10:1 compression
     (c) Size after 50:1 compression
     (d) For a 2-hour movie at 24 fps, calculate total sizes
2. **Entropy Calculation**:
   - Calculate entropy for these sources:
     (a) A die roll (6 equally likely outcomes)
     (b) Weather: Sunny(0.6), Cloudy(0.3), Rainy(0.1)
     (c) Binary source: P(0)=0.99, P(1)=0.01
   - Which is most compressible? Why?
3. **Real-world Analysis**:
   - Take three files from your computer: a .txt document, a .jpg image, and a .zip file

- Record their sizes
- Compress them using gzip at maximum compression
- Calculate compression ratios
- Explain why they compress differently

4. **Huffman Coding Practice**:

   - For the message "MISSISSIPPI":
     (a) Calculate symbol frequencies
     (b) Build Huffman tree
     (c) Assign codes
     (d) Encode the message
     (e) Calculate compression ratio vs 8-bit ASCII
     (f) Compare to entropy limit

5. **Research and Analysis**:

   - Find a current research paper on neural compression
   - Summarize its approach in 200 words
   - Compare its claimed performance to traditional methods
   - Identify one advantage and one limitation